

Structuri de date și algoritmi

Domeniul de studii: Ingineria Sistemelor / Specializarea: Automatică și Informatică Aplicată

SDA – Cursul 3

Conf. dr.ing. Adriana ALBU

adriana.albu@upt.ro

<http://www.aut.upt.ro/~adrianaa>

1. Structuri de date fundamentale (partea a doua)

Rezumat pentru cursul precedent

Tip de date abstract (TDA)
asociere între un model matematic (MM) și un set de operatori specifici

Tip de date (TD)
implementare a unui TDA într-un limbaj de programare

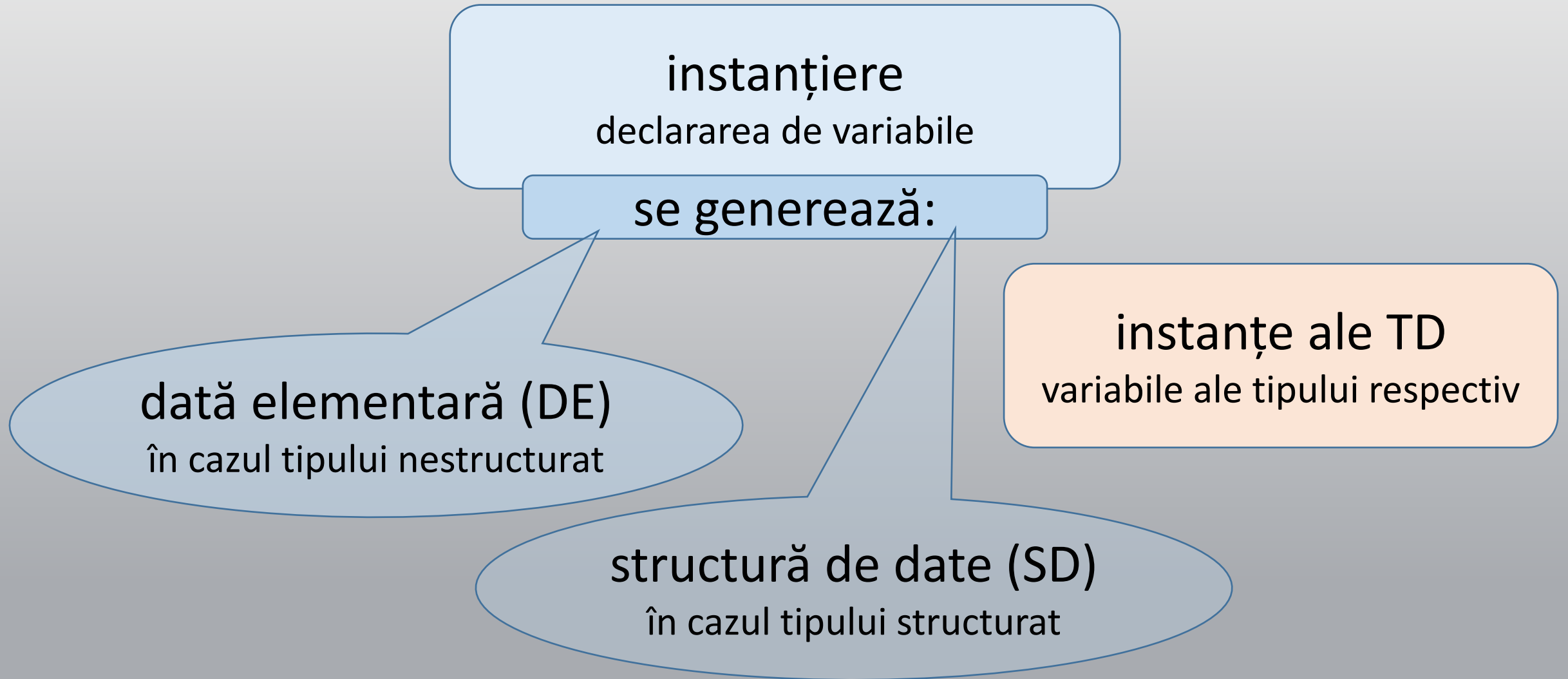
se caracterizează prin:

- mulțimea valorilor (pe care le pot lua elementele tipului respectiv)
- un anumit grad (nivel) de structurare (organizare) a informației
- set de operatori specifici

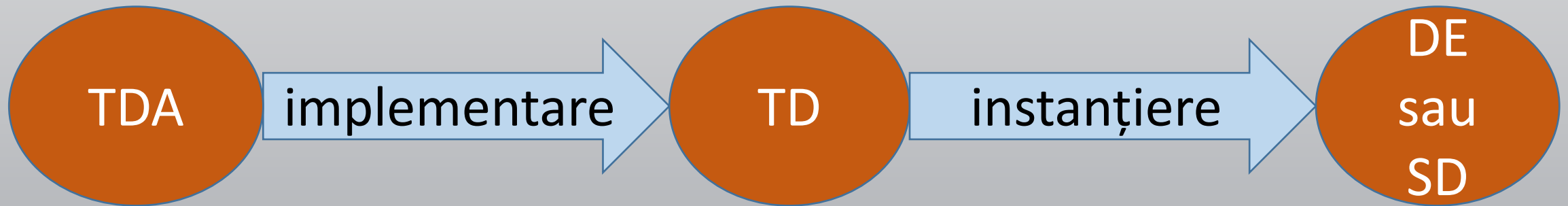
pot fi:

- nestructurate
- structurate

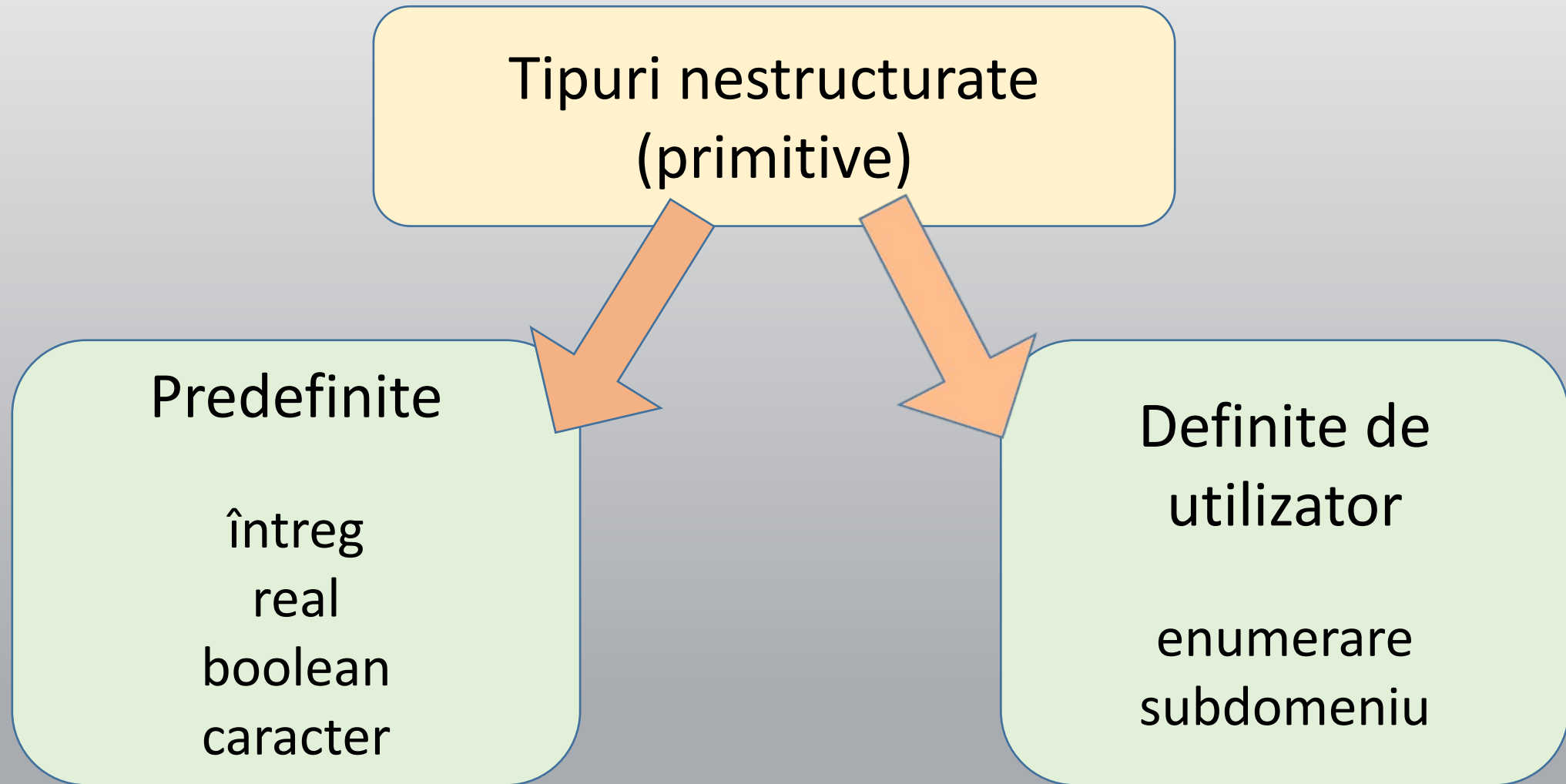
Rezumat pentru cursul precedent



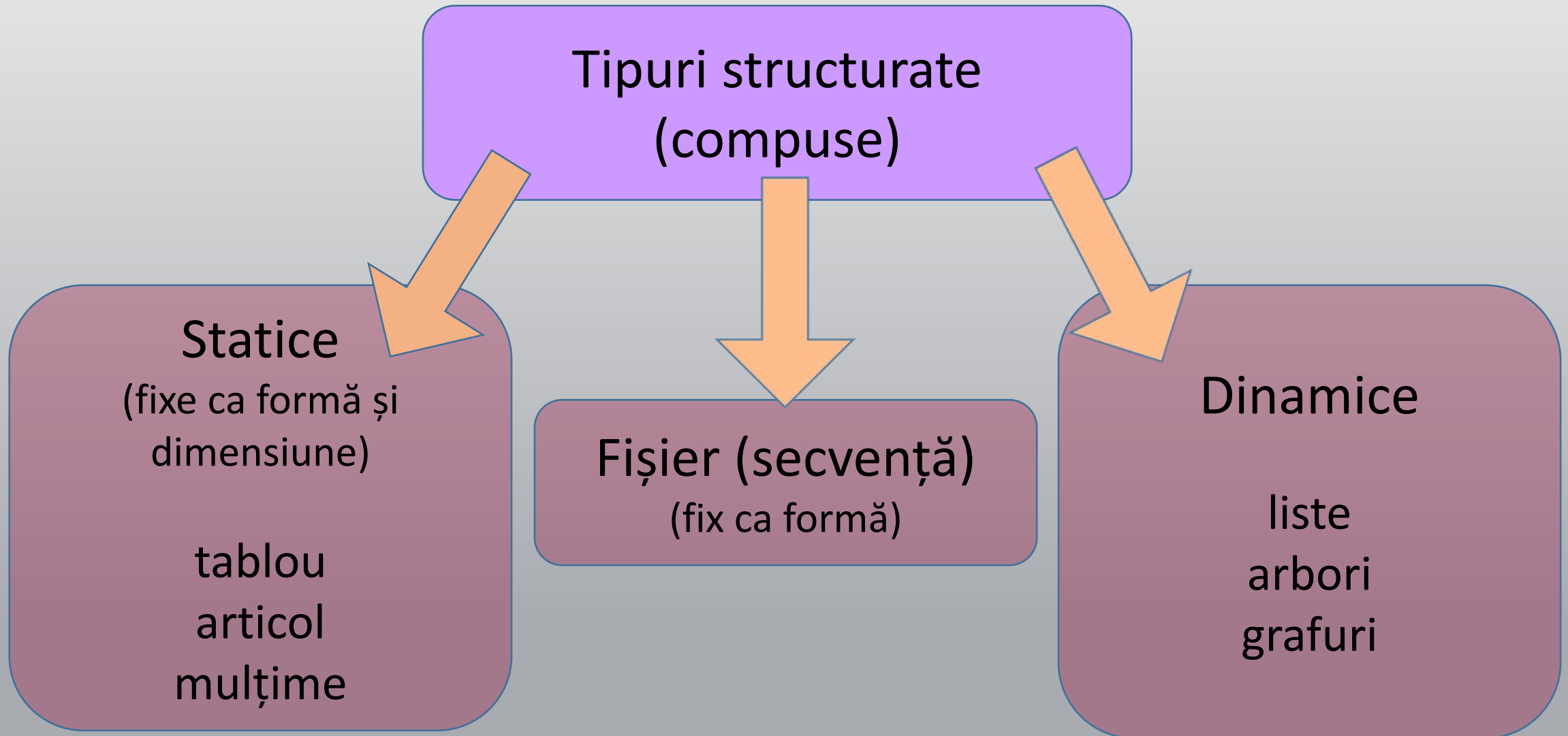
Rezumat pentru cursul precedent



Rezumat pentru cursul precedent



Rezumat pentru cursul precedent



1.4.3 Structura de date articol. TDA articol

- Se obține prin agregare, respectiv prin **reuniunea unor elemente aparținând mai multor tipuri constitutive** (structurate la rândul lor), într-un tip complex, structurat, numit **articol** (înregistrare)
- Mulțimea de valori asociată unui tip articol: totalitatea combinațiilor posibile ale valorilor tipurilor constitutive, selectând câte o singură valoare din fiecare tip

```
typedef struct{  
    tip1 nume1;  
    tip2 nume2;  
    ...  
    tipn numen;  
}articol;  
articol a;
```


1.4.3 Structura de date articol. TDA articol

- Articolul – structură de date **neomogenă**
- Selecția componentelor se realizează prin **identificatori de câmp** precizați la definirea tipului și **operatori de selecție**

```
articol a, *pa;
```

```
articol tab[N];
```

```
a.numel=...;
```

```
pa=tab; //pa=&tab[0];
```

```
pa->numel=...;
```

```
tab[0].numel=...;
```

- Selecția prin identificatori de câmp (nume de câmp) este o **selecție fixă**, realizată în urma unor elemente cunoscute încă în faza de compilare

TDA articol

➤ MM:

- colecție finită de elemente numite câmpuri, care pot să aparțină unor TD diferite
- există o corespondență biunivocă între lista identificatorilor de câmpuri și colecția de elemente

➤ Notatii:

- a – instanță a tipului articol
- id – identificator de câmp
- e – valoare a tipului asociat lui id

➤ Operatori:

- $DepuneArticol(a, id, e)$
- $FurnizeazaArticol(a, id) \rightarrow e$

Articol cu variante

- Se utilizează când este necesar ca două sau mai multe tipuri de date să fie definite ca variante ale aceluiași tip

```
typedef enum{NOTA,CALIFICATIV}tip_e;
typedef struct{
    char materie[20];
    tip_e evaluare; //camp selector
    union{
        float n;
        char c;
    }rezultat;
}examen;
examen e;
e.evaluare=NOTA;
e.rezultat.n=9.5;
```

- Pentru identificarea variantei curente s-a introdus **discriminatorul de tip** sau **câmpul selector**
- Mulțimea valorilor tipului articol cu variante
- rezultă din reuniunea tipurilor constitutive
 - cardinalitatea tipului articol este suma cardinalităților tipurilor constitutive

1.4.4 Structura de date mulțime. TDA mulțime

➤ Este un tip structurat fundamental, definit sau nu prin construcții sintactice:

```
TYPE TipMultime = SET OF T0;
```

```
TYPE indice = 1..10;
```

```
TYPE multimeIndice = SET OF indice;
```

- $2^{10} = 1024$ valori
- [], [1], [2], ..., [10], [1,2], [1,3], ..., [1,2,3,4,5,6,7,8,9,10]

- T_0 – tip de bază

- mulțimea valorilor lui T_0 – mulțime de bază

- mulțimea de valori pentru TipMultime = puterea mulțimii de bază

$\text{card}(\text{TipMultime}) = 2^{\text{card}(T_0)}$

➤ Fiecare valoare din mulțimea de bază este „prezentă” sau „absentă” în fiecare valoare asociată tipului mulțime

➤ O valoare a unei variabile de tip mulțime poate fi construită:

- Static – prin asignarea variabilei cu o constantă a tipului

- Dinamic – prin asignarea variabilei cu o expresie de calcul având drept operanzi mulțimi încadrate în același tip de bază

TDA mulțime

➤ MM: elementele tipului mulțime

- aparțin unui tip ordonat finit
- sunt membre ale unei mulțimi matematice

➤ Notatii:

- TipElement – tip de bază
- S, T, V – mulțimi ale căror elemente aparțin lui TipElement
- e – valoare a lui TipElement
- b – valoare booleană

➤ Operatori:

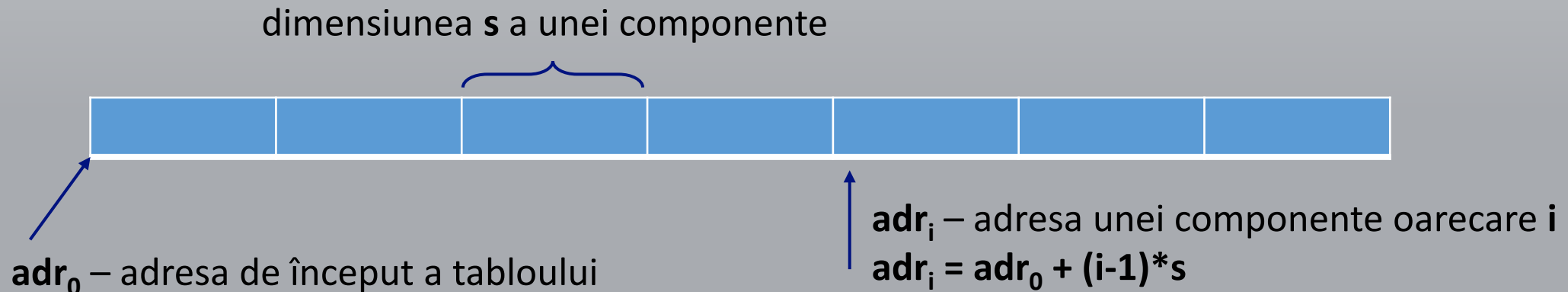
- specifici (legi de compoziție): atribuire, reuniune, scădere, intersecție;
- relaționali: egalitate, inegalitate, incluziune etc.
- DepuneMultime(S, T); EgalitateMultime(S, T)-> b ; ApartineMultime(S, e)-> b ;
Submultime(S, T)-> b ; Reuniune(S, T)-> V ; Intersectie(S, T)-> V

1.4.5 Reprezentarea structurilor de date abstracte

- Un program poate fi conceput, realizat și verificat pornind de la principiile care stau la baza nivelului de abstractizare utilizat
 - Utilizatorul (programatorul) este astfel eliberat de detaliile legate de implementarea conceptelor abstracte la nivel fizic
- **Reprezentare**
 - exprimarea structurilor abstracte în termenii tipurilor de date standard, implementate pe sistemul de calcul
 - la nivelul reprezentării se realizează corespondența dintre structura de date abstractă și memoria fizică
- **Memoria**
 - tablou format din celule individuale numite locații
 - indicii acestor locații se numesc adrese
 - dimensiunea memoriei – dată de cardinalitatea tipului
 - locațiile de memorie sau multiplii acestora formează unități de informație (de memorie)

Reprezentarea tablourilor

- Reprezentarea în memorie a unui tablou cu elemente de tip T
 - se realizează prin transformarea într-un tablou având drept componente unități de informație, situate într-o zonă contiguă de memorie
- Adresa $\mathbf{adr_i}$ asociată componentei i a tabloului se poate afla dacă se cunosc:
 - adresa de început $\mathbf{adr_0}$ (adresa de bază)
 - dimensiunea s în unități de memorie (biți, octeți) a componentelor
$$\mathbf{adr_i = adr_0 + (i-1)*s}$$

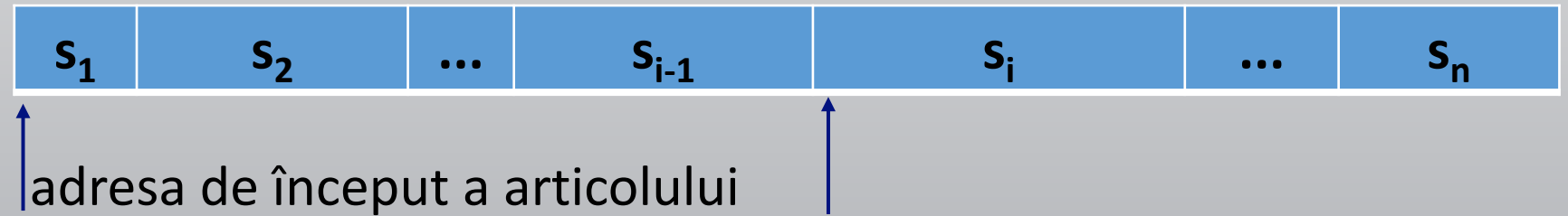


Reprezentarea articolelor

- Metoda de structurare fiind prin agregarea unor componente de tipuri diferite conduce la un calcul al adresei fiecărei componente funcție de următoarele elemente:
 - adresa de început a articolului
 - distanța relativă (deplasament, offset, adresă relativă) k_i a componentei i față de începutul articolului, măsurată în unități de informație
 - s_j = dimensiunea în unități de informație a unei componente j
 - $\Rightarrow k_i = s_1 + s_2 + s_3 + \dots + s_{i-1}$
- Identificatorii componentelor, cunoscuți încă în faza de compilare
 - reprezintă în fapt deplasamentele (adresele relative) asociate fiecărei componente în cadrul articolului

Reprezentarea articolelor

```
typedef struct{
    tip1 nume1;
    tip2 nume2;
    ...
    tipi numei;
    ...
    tipn numen;
} articol;
articol a;
```



k_i – adresa relativă față de începutul articolului a unei componente i
 $k_i = s_1 + s_2 + \dots + s_{i-1}$

Reprezentarea mulțimilor

➤ Mulțimea se reprezintă în memorie printr-o construcție numită **funcție caracteristică** C_m reprezentând un tablou unidimensional de valori logice în care componenta i va specifica prezența sau absența valorii i în mulțime

➤ $C_m = \text{card}(T_0)$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |

➤ Reprezentarea valorilor logice ca bit

- avantajează implementarea operatorilor pentru tipul mulțime prin suportul hardware (funcții logice, operații pe biți SI, SAU, deplasări)
- conduce la restricții în raport cu cardinalitatea tipului mulțime, funcție de lungimea vectorului binar utilizat în reprezentare

1.4.6 Structura de date secvență. TDA secvență

- Structurile de date prezentate (tablou, articol, mulțime)
 - sunt structuri de date statice (dimensiune de memorie fixă, cunoscută încă în faza de compilare) și au cardinalitate finită
- Structura de date secvență și structurile dinamice (liste, arbori, grafuri)
 - nu se pot încadra într-o dimensiune finită a cardinalității și necesită o abordare specifică
- Definiția recursivă: O structură secvență, având tipul de bază T_0 , se definește
 - fie drept o secvență vidă
 - fie drept o concatenare a unei secvențe având tipul de bază T_0 cu o valoare a tipului T_0
- Notatii:
 - $S_0 = \langle \rangle$
 - $S_i = \langle S_{i-1}, s_i \rangle, s_i \in T_0$
- Fiecare valoare a tipului secvență conține un număr finit de componente
 - dar acest număr este nemărginit, putându-se construi o secvență mai lungă

1.4.6 Structura de date secvență. TDA secvență

➤ Datorită cardinalității infinite

- volumul de memorie necesar reprezentării nu poate fi cunoscut în timpul compilării
- e necesar un mecanism de **alocare dinamică** a memoriei în timpul execuției

➤ Dacă un limbaj de programare dispune de funcții sistem care permit:

- **alocarea** și eliberarea memoriei
- legarea (înlănțuirea) și referirea dinamică a componentelor

atunci cu ajutorul instrucțiunilor limbajului pot fi gestionate structuri dinamice (structuri avansate de date)

➤ Maniera de definire a structurii secvență o încadrează în categoria structurilor de date avansate

➤ Alegerea unei reprezentări potrivite, cât și a unui set specific de operatori, permite includerea acestei structuri de date în categoria structurilor fundamentale, având operatorii direct implementați în limbaj

Structura fișier secvențial (fișier)

- Este un caz particular de secvență rezultat în urma restrângerii setului de operatori astfel încât este posibil doar **accesul secvențial** la componentele structurii (secvenței)
- Caracteristici:
 - structură omogenă
 - numărul componentelor, numit și lungime a secvenței, nu este cunoscut
 - cardinalitate infinită (o manieră de implementare constă în înregistrarea secvenței pe suporturi de memorie externă, reutilizabilă)
 - structură ordonată; ordonarea elementelor este stabilită de ordinea în timp a adăugării componentelor
 - în fiecare moment este accesibilă o singură componentă, numită componentă curentă, precizată printr-un indicator (pointer) asociat secvenței
 - indicatorul (pointerul) avansează secvențial, după fiecare operație executată asupra secvenței (fișierului)
 - uzual i se asociază un operator care semnalează sfârșitul fișierului (EOF)

TDA secvență (fișier secvențial)

➤ MM

- secvență omogenă de elemente (unic tip constitutiv, numit și tip de bază)
- un indicator asociat secvenței, indică spre următorul element la care se poate realiza accesul

➤ Notatii

- TipElement – tip de bază (nu poate fi la rândul său secvență)
- f – secvență; e – valoare de TipElement; b – valoare booleană

➤ Operatori

- Rescrie(f) – poziționează indicatorul la începutul secvenței și pregătește secvența pentru scriere
- DepuneSecventa(f, e) – pentru o secvență deschisă în regim de scriere, operatorul copiază valoarea e în elementul următor al secvenței și avansează indicatorul
- EOF(f) -> b
- FurnizeazaSecventa(f, e) – returnează în e valoarea următorului element al secvenței și avansează indicatorul; secvența trebuie deschisă în regim de citire
- Adauga(f) – deschide secvența în regim de scriere, poziționând indicatorul la sfârșit

Vă mulțumesc!